

A PROJECT REPORT
ON
STEGANOGRAPHY

Project submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
BY

T K ABHILASH
RISHKA PASARAM
SAMUDRALA ANVESH

(18C91A0597)
(18C91A0582)
(18C91A0589)

Under the Esteemed guidance of
Mrs. B.V.Ramana M.Tech(Ph.D)
Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE
(COLLEGE OF ENGINEERING)

(Approved by AICTE New Delhi, Permanently Affiliated to JNTU Hyderabad, Accredited by NAAC with 'A' Grade)
Bogaram (V), Keesara (M), Medchal District -501 301.

2021 - 2022

HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE

(COLLEGE OF ENGINEERING)

*(Approved by AICTE New Delhi, Permanently Affiliated to JNTU Hyderabad, Accredited by NAAC with 'A' Grade)
Bogaram (V), Keesara (M), Medchal Dist-501301.*

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the mini project entitled “ **STEGANOGRAPHY** ” is being submitted by **T K ABHILASH(18C91A0597), RISHIKA PASARAM (18C91A0582),SAMUDRALA ANVESH (18C91A0589)**, in Partial fulfillment of the academic requirements for the award of the degree of Bachelor of Technology in “**COMPUTER SCIENCE AND ENGINEERING**” **HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE, JNTU Hyderabad** during the year 2019- 2020.

INTERNAL GUIDE

Mrs.B.V.Ramana M.Tech(Ph.D)
Professor
Dept. of Computer Science & Engineering.

HEAD OF THE DEPARTMENT

DR .B.NARSIMHA M.Tech, Ph.D.
Professor & HoD
Dept. of Computer Science & Engineering

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, who's constant guidance and encouragement crowns all effort with success.

I take this opportunity to express my profound gratitude and deep regards to My Guide **Mrs.B.V. Ramana, Professor**, Dept. of Computer Science & Engineering, Holy Mary Institute of Technology & Science for his / her exemplary guidance, monitoring and constant encouragement throughout the project work.

My special thanks to **Dr. B. Narsimha, Head of the Department**, Dept. of Computer Science & Engineering, Holy Mary Institute of Technology & Science who has given an immense support throughout the course of the project.

I also thank to **Dr. P. Bhaskara Reddy**, the **honorable Director** of my college Holy Mary Institute of Technology & Science for providing me the opportunity to carry out this work.

At the outset, I express my deep sense of gratitude to the beloved **Chairman A. Siddhartha Reddy of Holy Mary Institute of Technology & Science**, for giving me the opportunity to complete my course of work

I am obliged to **staff members** of Holy Mary Institute of Technology & Science for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Last but not the least I thank **ALMIGHTY** and My **Parents**, and **Friends** for their constant encouragement without which this assignment would not be possible.

T K ABHILASH

(18C91A0597)

RISHIKA PASARAM

(18C91A0582)

SAMUDRALA ANVESH

(18C91A0589)

DECLARATION

This is to certify that the work reported in the present project titled “**STEGANOGRAPHY** ” is a record of

work done by me in the Department of Computer Science & Engineering, Holy Mary Institute of Technology and Science.

No part of the thesis is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text the reported are based on the project work done entirely by me not copied from any other source.

T K ABHILASH

(18C91A0597)

RISHIKA PASARAM

(18C91A0582)

SAMUDRALA ANVESH

(18C91A0589)

CONTENTS

Names of the chapter	page No.
1. ABSTRACT	
INTRODUCTION.....	6
PROJECT SCOPE	6
PURPOSE.....	7
MOTIVATION	7
2. LITERTURE SURVEY	
INTRODUCTION.....	8
PROPOSED SYSTEM.....	8
EXISTING SYSTEM.....	8
3.SYSTEM ANALYSIS	
HARDWARE REQUIREMENT.....	9
SOFTWARE REQUIREMENT	9
STSKEHOLDER.....	10
CLR.....	10
4 . SYSTEM DESIGN	
SOFTWARE DESIGN.....	12
DATA FLOW DIAGRAM.....	12
USER CASE DIAGRAM.....	16

ACTIVITY DIAGRAM.....	17
CLASS DIAGRAM.....	18

5. IMPLEMENTATION

THE ENCODING.....	21
THE DECODING.....	21
SPATIAL METHOD.....	22
SYSTEM IMPLEMENTATION.....	25
SAMPLE CODE.....	35

6.SAMPLE TESTING

METHODS.....	42
TEST CASES.....	43

BIBLIOGRAPHY.....	45
CONCLUSION	46
REFERENCE.....	47

LITS OF FIGURES

Figure Names	page no.
PROCESS OF STEGANOGRAPHY.....	6
STAKE HOLDER.....	10
DFD LEVEL 0.....	13
DFD LEVEL 1.....	14
DFD LEVEL 2.....	15
USER CASE DIAGRAM.....	16
ACTIVITY DIAGRAM.....	17
CLASS DIAGRAM.....	18
E-R DIAGRAM.....	19
IMPLEMENTATION.....	20
SCREENSHOTS OF SYSTEM.....	25
RESULT.....	38
TEST CASES.....	41

STEGANOGRAPHY

ABSTRACT

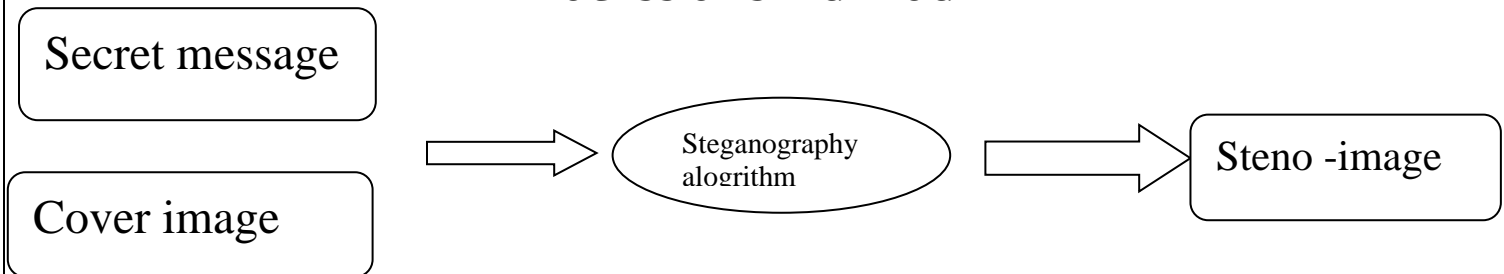
Security often requires that data be kept safe from unauthorized access. And the best line of defense is physical security. However, physical security is not always an option. Instead, most computers are interconnected with each other openly, thereby exposing them and the communication channels that they use. Steganography secures information by protecting its confidentiality. It can also be used to protect information about the integrity and authenticity of data. So, far Steganography is used in many forms but using it with Audio, Video & Image files is another Stronger Technique. The process of Steganography happens with Image File for transferring more secure sensitive data.

1.INTRODUCTION

WHAT IS STEGANOGRAPHY?

Steganography is a Greek word which means concealed writing. The word stegan means covered and graphical means writing. Thus, steganography is not only the art of hiding data but also hiding the fact of transmission of secret data. Steganography hides the secret data in another file in such a way that only the recipient knows the existence of message. In ancient time, the data was protected by hiding it on the back of wax, writing tables, stomach of rabbits or on the scalp of the slaves. But today's most of the people transmit the data in the form of text, images, video, and audio over the medium. In order to safely transmission of confidential data, the multimedia object like audio, video, images are used as a cover source to hide the data. Steganography is defined as the study of invisible communication.

PROCESS OF STEGANOGRAPHY



PROJECT SCOPE:

Our project scope is developed for hiding information in any image file to ensure the safety of exchange the data between different parties and provide better security during message transmission.

The scope of the project is implementation of steganography tools for hiding information includes any type of information file and image files and the path where the user wants to save image and extruded file. We will use LSB technique; the proposed approach is to use the suitable algorithm for embedding the data in an image file; we will show a brief of this algorithm that we used to hiding data.

PURPOSE:

The purpose of this application is to provide the security for the confidential information. This application does allow the hackers to view the data, can view only Image file when it is being passed over the internet.

Then at the recipient side the original information i.e., plain text will be extracted from the Image by performing decryption operations.

MOTIVATION:

This application uses python to perform encryption and decryption operations. In the encryption process each first the data will be encrypted by the key which was given by the source and then this encrypted data will be embed into the Audio, Video & Image file and generate the new Audio, Video & Image file which contains the plain text.

At the recipient side this data will be extracted and decrypted then gives plain text.

2. LITERATURE SURVEY

INTRODUCTION:

Steganography aims to hiding information in a cover data in such a way that nonparticipating persons are not able to detect the presence of this information by analyzing the information detection. Unlike watermarking, steganography does not intend to prevent the hidden information by opponents of removing or changing the hidden message, which is embedded in the cover data but it emphasizes on remains it undetectable. Steganography is particularly interesting for applications in which the encryption cannot used to protect the communication of confidential information.

PROPOSED SYSTEM:

In the proposed system the above problem has been solved by embedding the data into the Image file. Before embedding it into the file, encryption operation will be performed by using the encryption key which is provided by the source.

Then this Image file will be passed over the net, even if hacker hacks it, can be able to see only a Image file. At the destination side this data will be encrypted from Audio, Video & Image file and performs decryption to get original message.

EXISTING SYSTEM:

If a person sends sensitive information over the insecure channels of the system, then there may be a chance of hacking it, they can alter the information and sends over the net. (Example is military persons sending sensitive information over the net.)

3.ANALYSIS

Hard Ware Requirements

- Processor: Pentium-III (or) Higher
- Ram: 4GB (or) Higher
- Hard disk: 10GB

Soft Ware Requirements

- Operating System: Windows
- Languages: PHYTHON

Design and implementation:

This project will be developed using the technologies like PYTHON. Also I'll be learning clearly about Software Development Life Cycle.

User document:

This document also includes a user manual which assists the new user to go about the project, he can even get the online help which caters the needs of a new user and makes this project more user friendly, a step by step approach online makes it easy to use software for a naïve user.

Stakeholders:

Stakeholders must be able to make decisions, but need not be human: "An stakeholder might be a person, a company or organization, a computer program, or a computer system-hardware, software, or both." Actors are always stakeholders, but not all stakeholders are actors. The following Table 3.1: Stakeholders and its Description.

Stakeholder	Description
User	That have major controlling of the whole systems process and able to access or use all modules of the application.
Program	The Application itself which can perform the user operation in main process encrypt (hide image in another image), decrypt (unhide image from Stego-image).

Table 3-1: Stakeholders

Choose an image :Select the picture, open the image, the image review.

Processing image :typing image in the image box pressing the button for the treatment process

Hide image :The program hides image.

A use case is a written description of how users will perform tasks on your website. It outlines, from a user's point of view, a system's behavior as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal The use case description of our project as follows;

CLR: (COMMON LANGUAGE RUNTIME)

As mentioned above, the "Common Language Runtime" is an important component of .NET platform. The CLR handles the object layouts and manages references to objects. It makes the designing architecture simpler and easier. I used Microsoft Visual C# for structuring the application. CLR is efficient when C# language is used for writing programs. The various advantages when the C# language is employed are:

- Complete Object oriented design.
- Provides high security and stronger to break the code.
- Uses the logics of visual basic and C++ languages
- The syntax is simple is similar to languages C. C++ (Microsoft, 2010, MSDN).

Windows Forms:

Visual Studio Windows application is supported by the Microsoft NET frame work and it allows a rich set of classes so that the Windows application can be built by any net programming language like Visual Basic .net and Visual C#.

Here, in this project used visual c # to create the windows-based application. The windows-based application differs with Web form application. In windows-based application, we create some type of forms which are used for user interface. These forms are known as "Windows forms".

Actor:

An actor in the Unified Modeling Language (UML) "specifies a role played by a user or any other system that interacts with the subject." "An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject.

There are two actors in our application:

1. User
2. Program

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use

4. SYSTEM DESIGN

System design is transition from a user-oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

The database tables are designed by analyzing functions involved in the system and format of the fields is also designed. The fields in the database tables should define their role in the system. The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

SOFTWARE DESIGN

In designing the software following principles are followed:

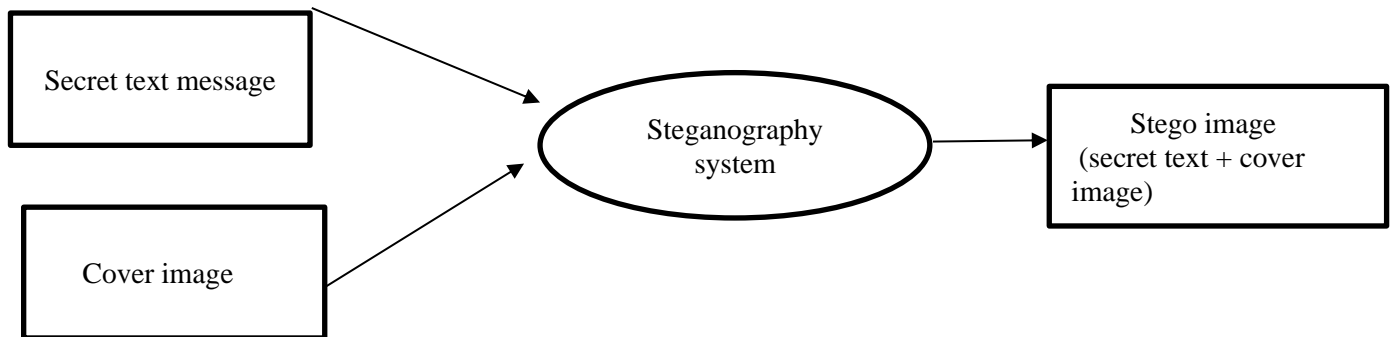
1. Modularity and partitioning: software is designed such that, each system should consist of hierarchy of modules and serve to partition into separate function.
2. Coupling: modules should have little dependence on other modules of a system.
3. Cohesion: modules should carry out in a single processing function.
4. Shared use: avoid duplication by allowing a single module is called by other that need the function it provides.

DATAFLOW DIAGRAMS

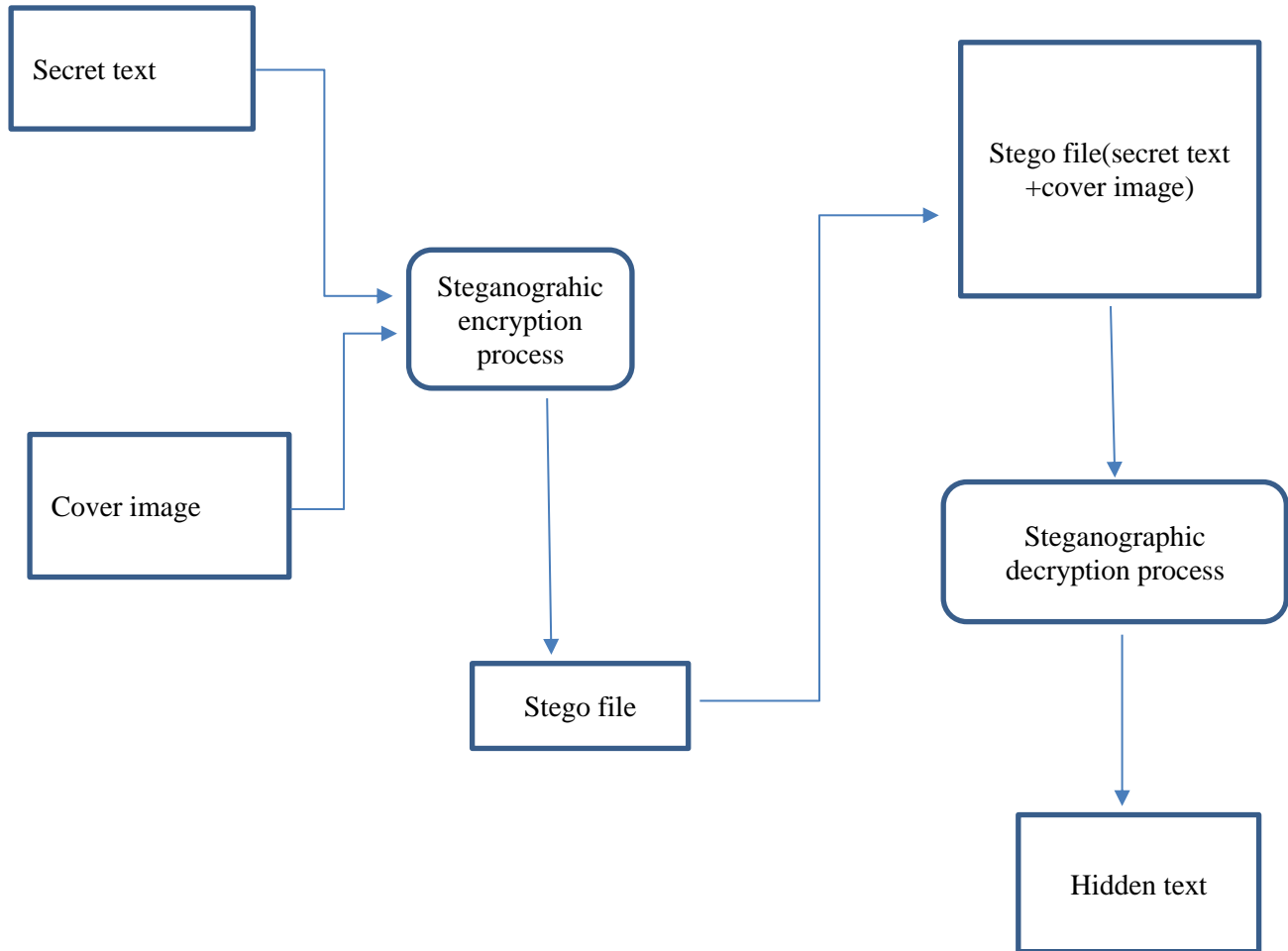
- Data flow diagram is a graphical tool used to describe analyze the movement of data through a system manual or automated including the processes, stores of data, and delays in the system.
- Data flow diagrams are the central tool and basis for form which other components are developed. The data flow diagram is also known a data flow graph or bubble chart.

DFD LEVEL 0

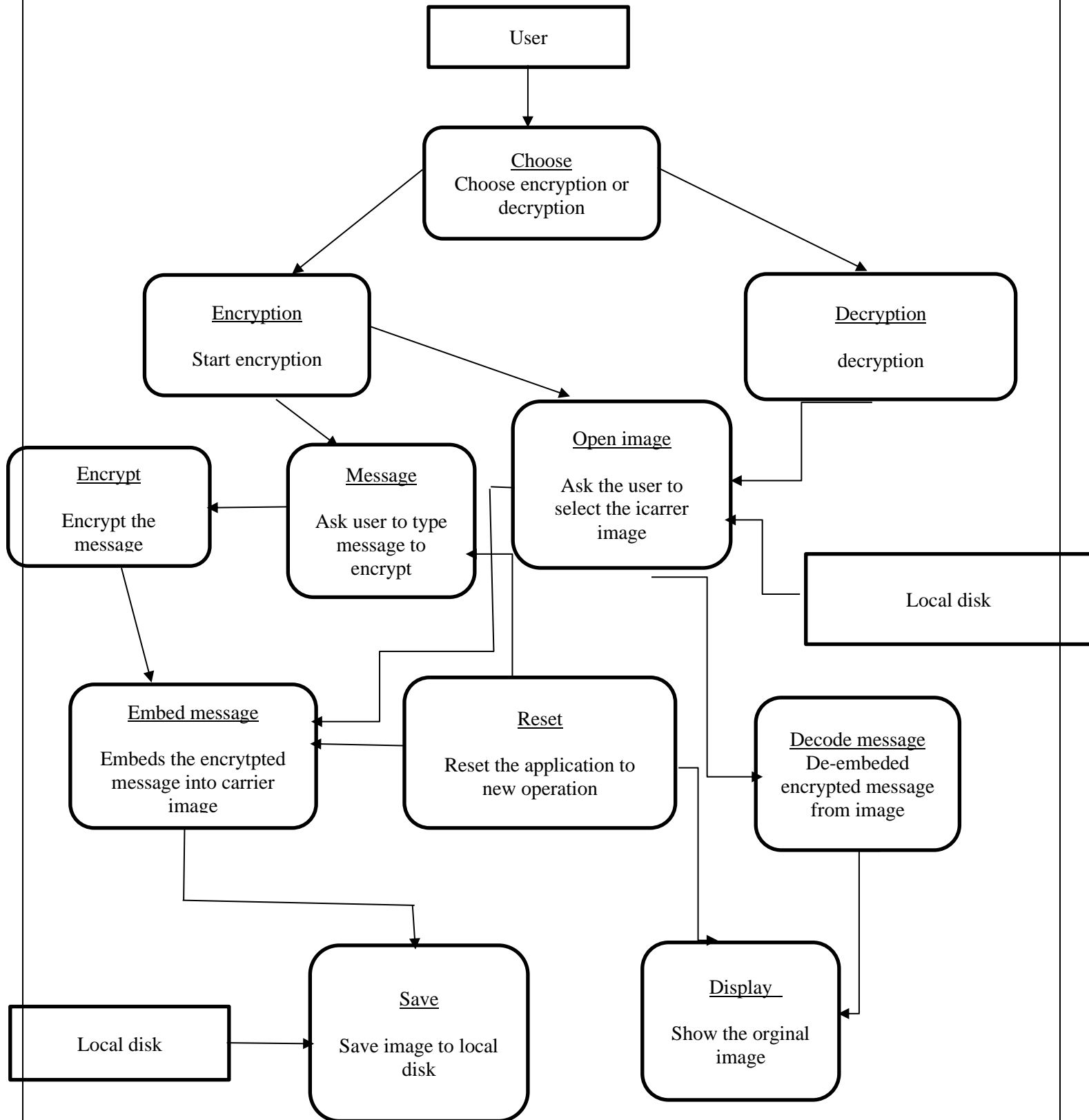
Steganography system takes inputs as secret text message as well as cover image and generates steno image.



DFD LEVEL 1

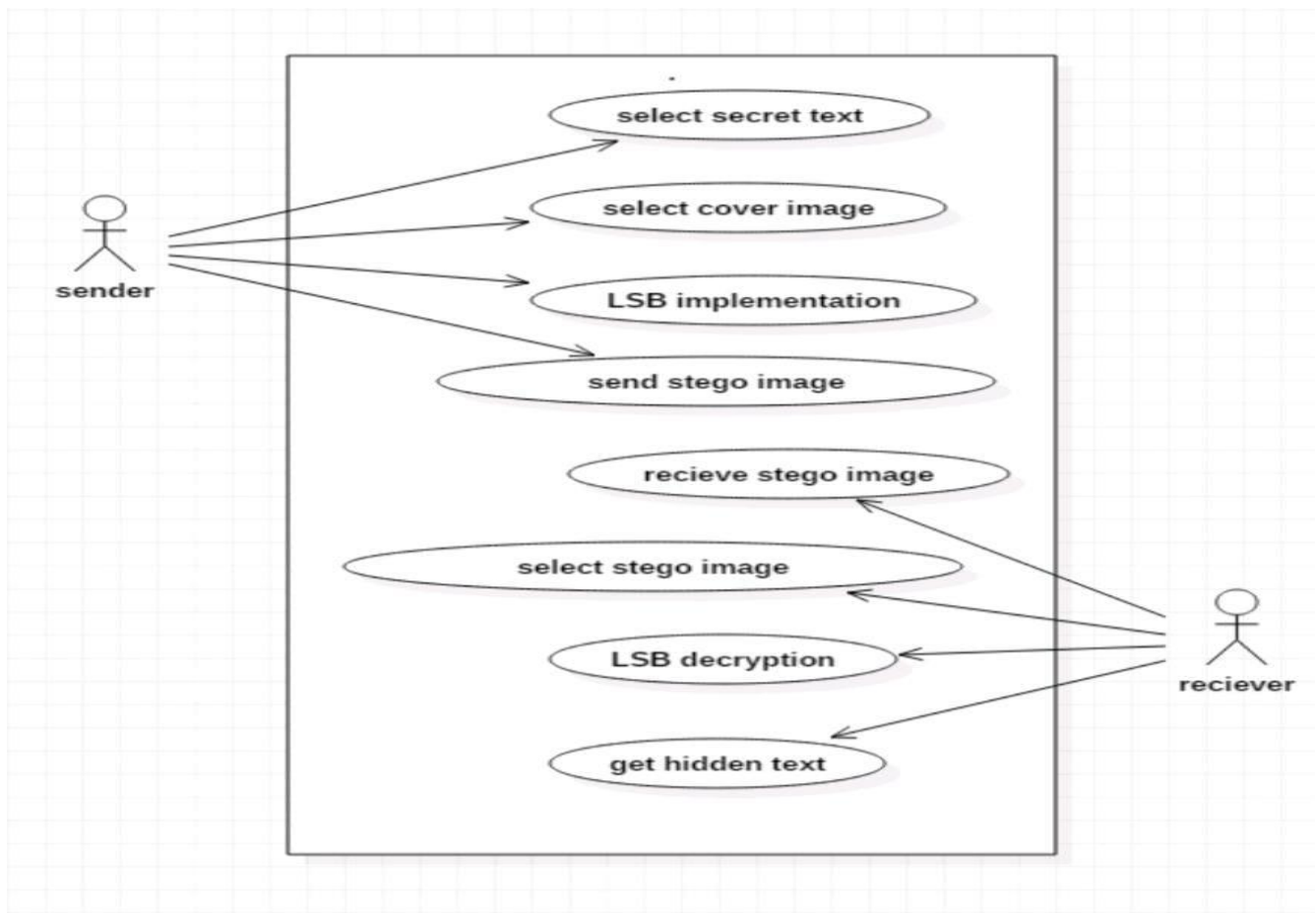


DFD LEVEL 2



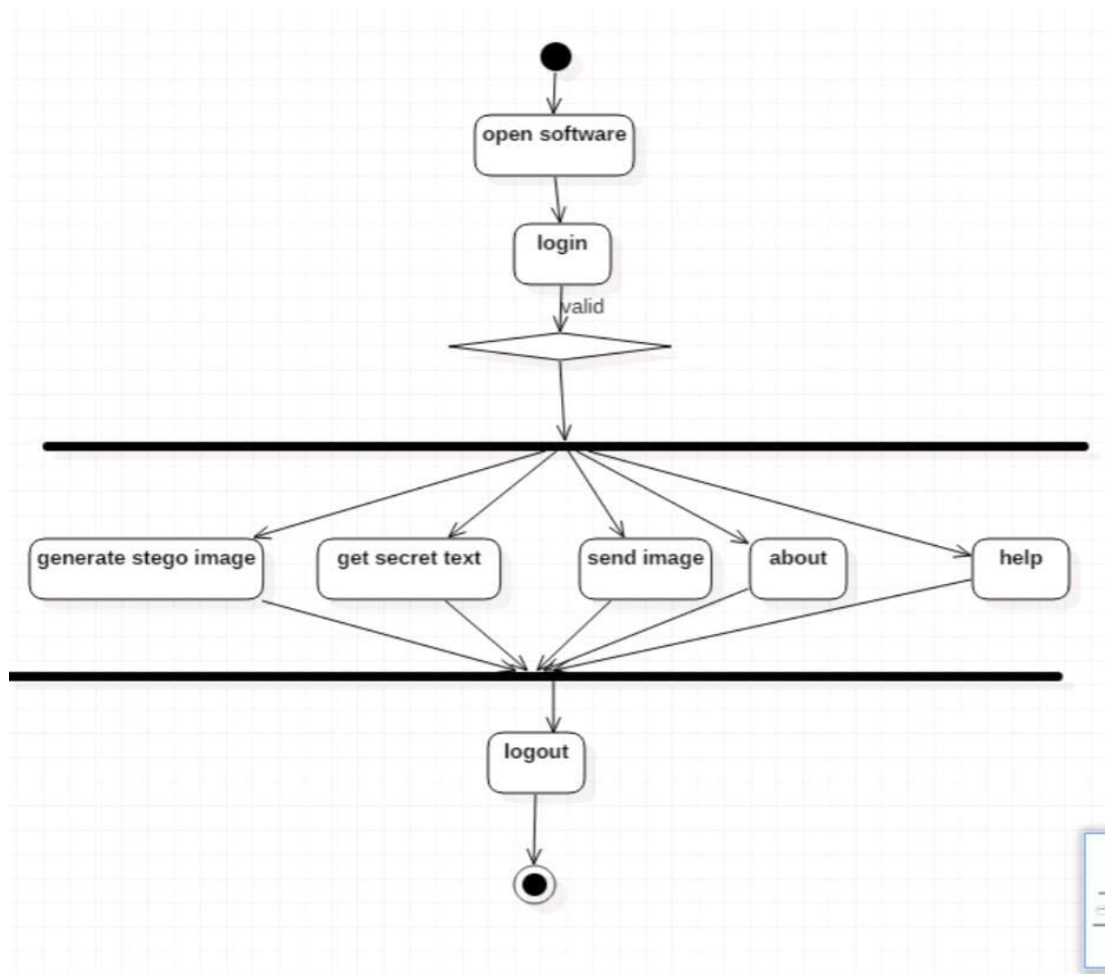
USECASE DIAGRAM

A Use Case Diagram at its simplest is a representation of a user's interaction with the system. First user writes secret text then he selects cover image and data gets hidden inside image, then user sends steno image to receiver through image. At the receiver side, user selects the steno image and applies decryption on steno image. After that he can get text hidden in the text.



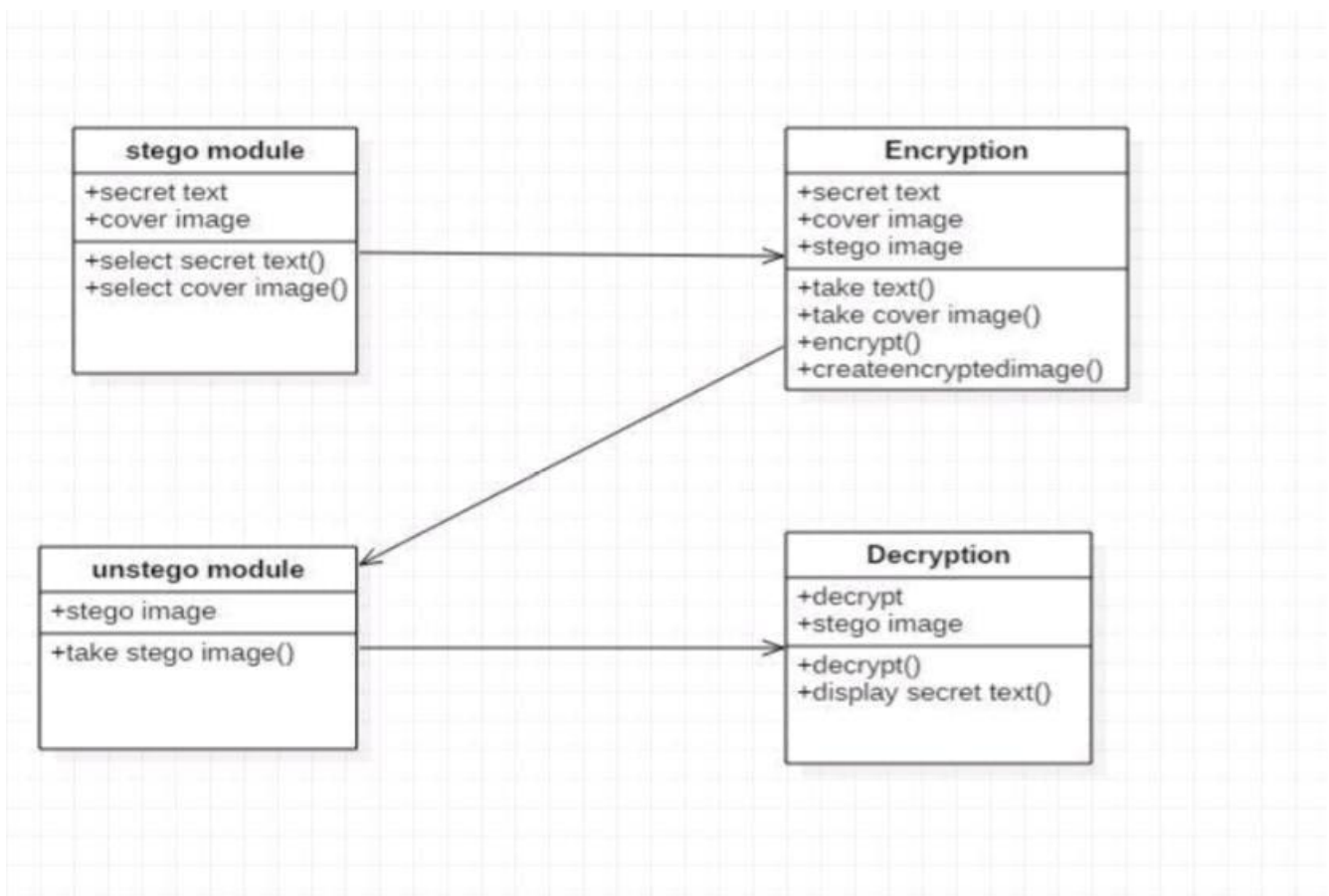
ACTIVITY DIGRAM

Purpose: An example of UML activity diagram describing behavior of the Steno System for Secure Communication.

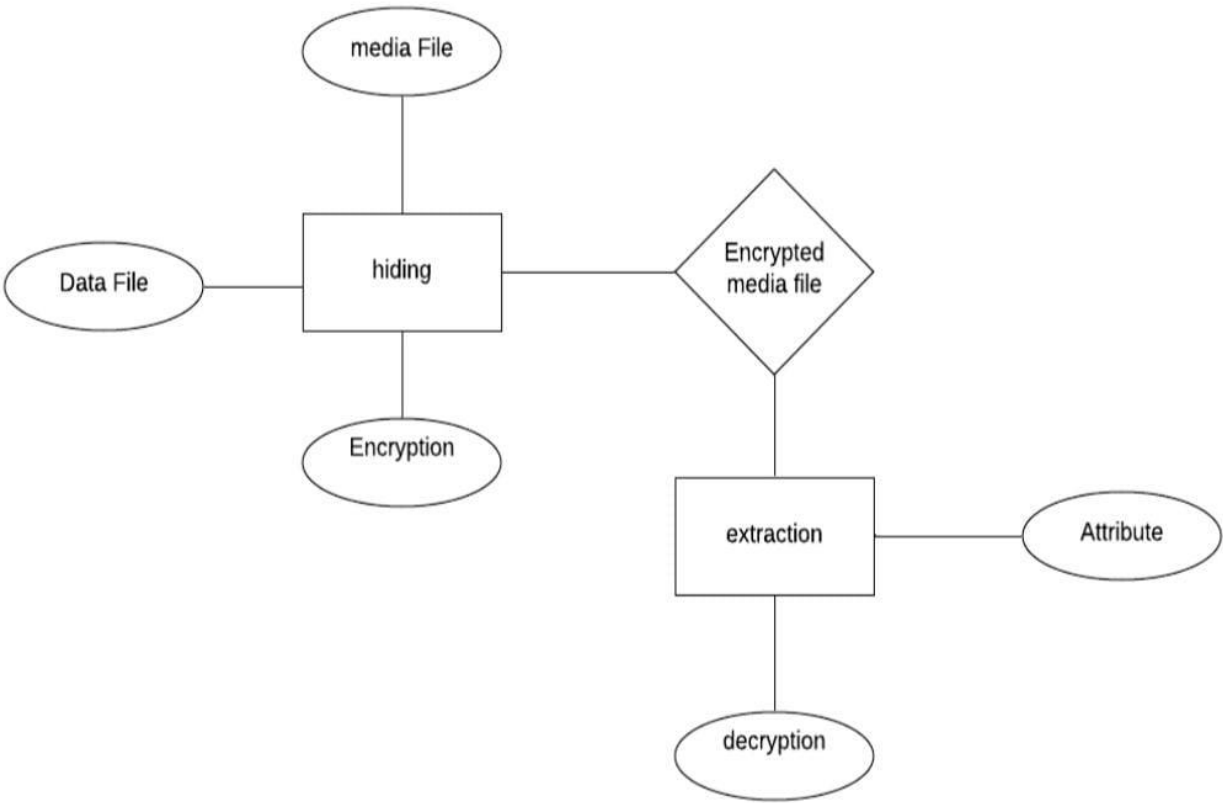


CLASS DIAGRAM

- **Steno module:** secret module consists of two attributes secret text and cover image as well as two operations as select secret text() and select cover image()
- **Encryption:** Encryption is the process of hiding secret text into cover image.
- **Ustego Module:** It takes input from steno image that is steno image.
- **Decryption:** It separates the secret text from the image.



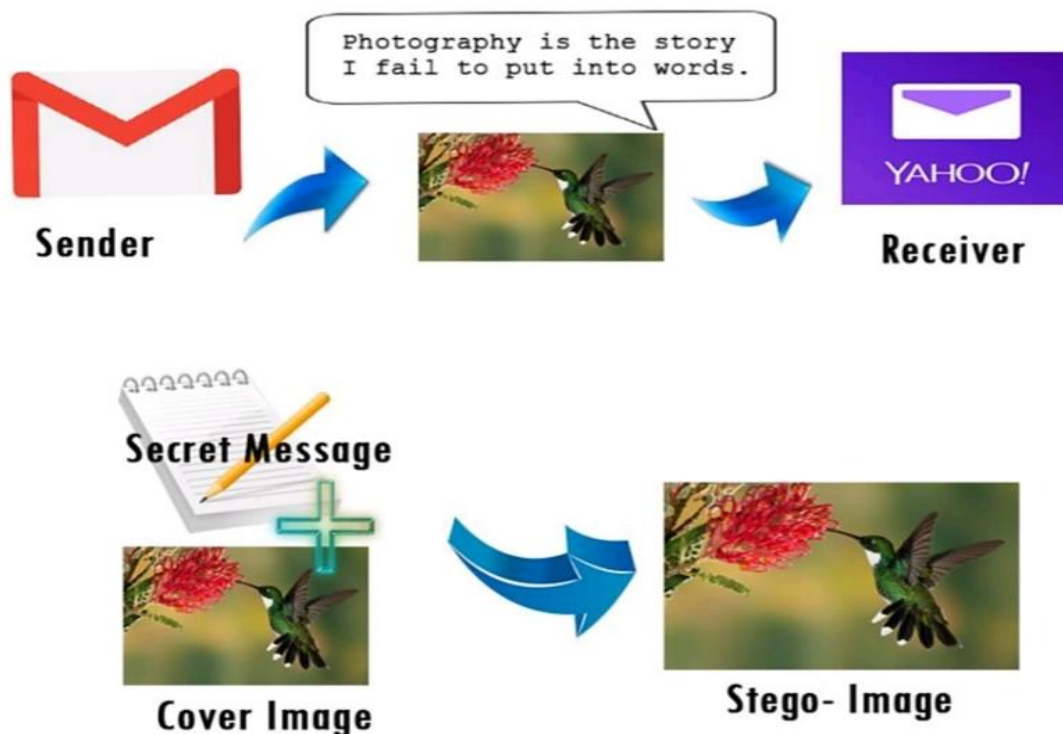
E-R DIAGRAM



5.IMPLEMENTATION

Hiding information inside images is a popular technique nowadays. An image with a secret message inside can easily be spread over the World Wide Web or in newsgroups. The use of steganography in newsgroups has been researched by German steganographic expert Niels Provos, who created a scanning cluster which detects the presence of hidden messages inside images that were posted on the net.

However, after checking one million images, no hidden messages were found, so the practical use of steganography still seems to be limited. Image Steganography is the technique of hiding the data within the image in such a way that prevents the unintended user from the detection of the hidden messages or data.



COMMUNICATION THROUGH STEGANOGRAPHY

To hide a message inside an image without changing its visible properties, the cover source can be altered in noisy areas with many color variations, so less attention will be drawn to the modifications. The most common methods to make these alterations involve the usage of the least-significant bit or LSB, masking, filtering and transformations on the cover image. These techniques can be used with varying degrees of success on different types of image files. The project deals with learning about the various types of steganography available.

Image steganography is performed for images and the concerning data is also decrypted to retrieve the message image. Since this can be done in several ways, image steganography is studied and one of the methods is used to demonstrate it. Image steganography refers to hiding information i.e., text, images or audio files in another image or video files.

The current project aims to use steganography for an image with another image using spatial domain technique. This hidden information can be retrieved only through proper decoding technique. This encryption and decryption of the images is done using PYTHON.

IMPLEMENTATION

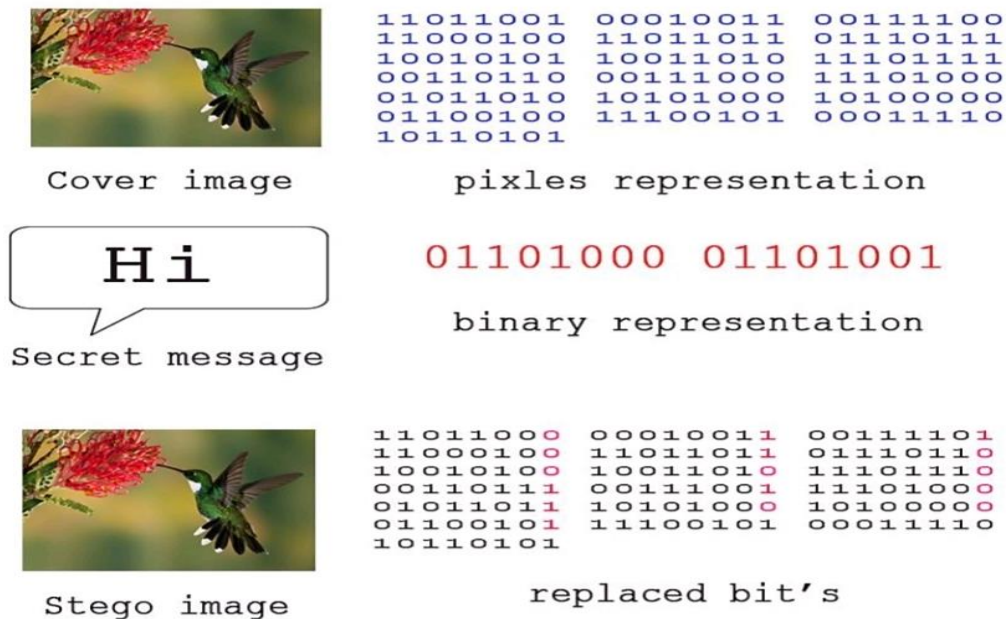
THE ENCODING

The steganography technique used is LSB coding. The offset of the image is retrieved from its header. That offset is left as it is to preserve the integrity of the header, and from the next byte, we start our encoding process. For encoding, we first take the input carrier file i.e. an image file and then direct the user to the selection of the text file.

THE DECODING PROCESS

The data of image is taken into byte array. Using above byte array, the bit stream of original text file is retrieved into another byte array. And above byte array is written into the decoded text file, which leads to the original message.

LSB (least significant bit)



SPATIAL METHOD

In spatial method, the most common method used is LSB substitution method. Least significant bit (LSB) method is a common, simple approach to embedding information in a cover file.

In steganography, LSB substitution method is used. I.e., since every image has three components (RGB). This pixel information is stored in encoded format in one byte. The first bits containing this information for every pixel can be modified to store the hidden text.

For this, the preliminary condition is that the text to be stored has to be smaller or of equal size to the image used to hide the text.

LSB based method is a spatial domain method. But this is vulnerable to cropping and noise. In this method, the MSB (most significant bits) of the message image to be hidden are stored in the LSB (least significant bits) of the image used as the cover image.

It is known that the pixels in an image are stored in the form of bits.

In a grayscale image, the intensity of each pixel is stored in 8 bits (1byte). Similarly, for a color (RGB-red, green, blue) image, each pixel requires 24 bits (8bits for each layer).

The Human visual system (HVS) cannot detect changes in the color or intensity of a pixel when the LSB bit is modified. This is psycho-visual redundancy since this can be used as an advantage to store information in these bits and yet notice no major difference in the image.

Algorithm of LSB method of steganography. There might be two different phases of LSB method, embedding phase and extracting phase. Algorithms of both of the phases are given below:

EMBEDDING PHASE PROCEDURE:

Step 1: Extract all the pixels from the given image and store them in some array named (image array).

Step 2: Extract all the characters from the given text file (message file) and store it in the array called (message array).

Step 3: Retrieve the characters from the Stego key and store them in a array called Key array. A steno-key is used to control the hiding process so as to restrict detection and/or recovery of the embedded data.

Step 4: Take first pixel and characters from Key- array and place it in first component of pixel. If there are more characters in Key array, then place rest in the first component of next pixels.

Step 5: Place some terminating symbol to indicate end of the key. 0 has been used as a terminating symbol in this algorithm.

Step 6: Place characters of message Array in each component of next pixels by replacing it.

Step 7: Repeat step 6 till all the characters has been embedded.

Step 8: Again, place some terminating symbol to indicate end of data.

Step 9: Obtained image will hide all the characters that input. The simplest steganography techniques embed the bits of the message directly into least significant bit plane of the cover image in a deterministic sequence. Modulating the least significant bit does not result in human-perceptible difference because the amplitude of the change is small.

To hide a secret message inside an image, a proper cover image is needed. Because this method uses bits of each pixel in the image, it is necessary to use a lossless compression format, otherwise the hidden information will get lost in the transformations of a lossy compression algorithm.

When using a 24-bit color image, a bit of each of the red, green and blue color components can be used, so a total of 3 bits can be stored in each pixel. For example, the following grid can be considered as 3 pixels of a 24-bit color image, using 9 bytes of memory:

(00100111 11101001 11001000)

(00100111 11001000 11101001)

(11001000 00100111 11101001)

When the character A, which binary value equals 10000001, is inserted, the following grid results:

(00100111 11101000 11001000)

(00100110 11001000 11101000)

(11001000 00100111 11101001)

In this case, only three bits needed to be changed to insert the character successfully. On average, only half of the bits in an image will need to be modified to hide a secret message using the maximal cover size.

The result changes that are made to the least significant bits are too small to be recognized by the human visual system (HVS), so the message is effectively hidden. As you see, the least significant bit of third color is remained without any changes.

It can be used for checking the correctness of 8 bits which are embedded in these 3 pixels. In other words, it could be used as parity bit.

SCREENSHOTS OF SYSTEM:



Figure 5-1: Embed File - Encryption Phase

To start hiding a file you must the following steps:

1. Select the Embed from the tabs.
2. Select the algorithm type.
3. Select the cover file by click browse until the open file dialog "Choose a Cover" file appears.
4. From the "Choose a Cover File" dialog choose the cover file and click open.
5. Select the embed file by click browse until the open file dialog "Choose a File To Embed" appear.
6. From the " Choose a File To Embed " dialog choose the embed file and click open.
7. Select the output file by click browse until the save file dialog "Save as .." appear.
8. Select the path where you want to save the file and type the file name then click save.
9. Finally click on the embed button to hiding the embed file information inside the cover file.
10. After the hiding data, message box will appear to tell user the hiding operation is successfully.

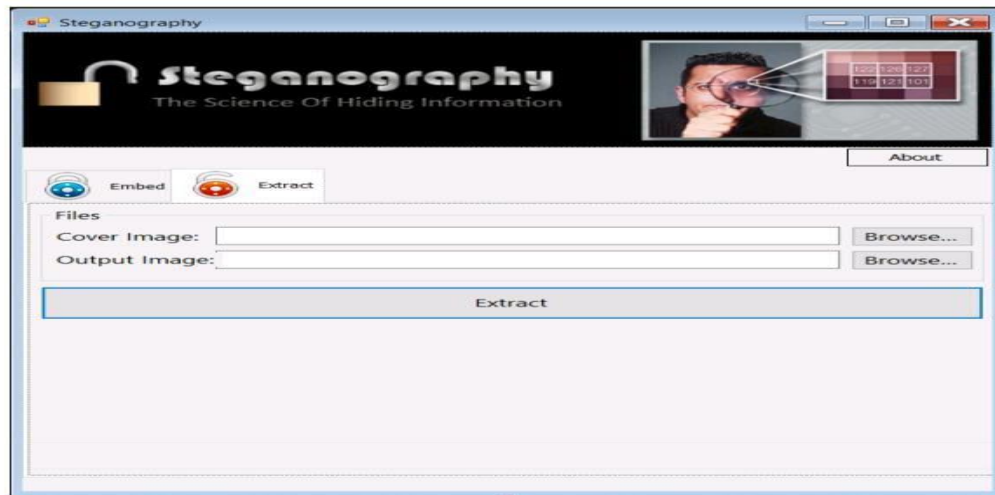


Figure 5-2: Extract File - Decryption Phase

To start extracting a file you must the following steps:

1. Select the Extract from the tabs.
2. Select the algorithm type.
3. Select the cover file by click browse until the open file dialog "Choose a Cover" file appears.
4. From the "Choose a Cover File" dialog choose the cover file and click open.
5. Select the output file by click browse until the save file dialog "Save as .." appear.
6. Select the path where you want to save the file and type the file name then click save.
7. Finally click on the extract button to extracting the embed file information.

8. After the extracting data, message box will appear to tell user the extracting operation is successfully.

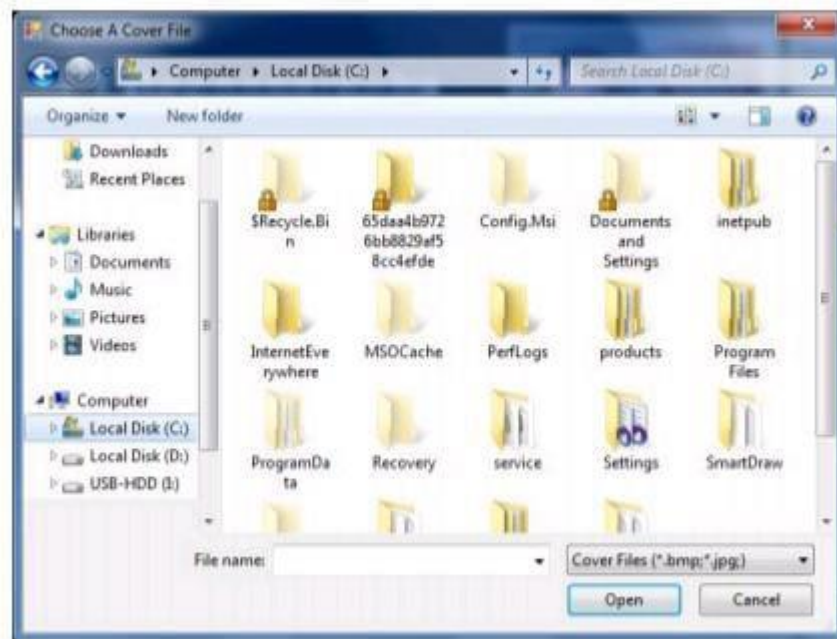


Figure 5-3: Choose a cover file dialog

From this screen you must select the cover file by choose the right path and select the file and then click the open button

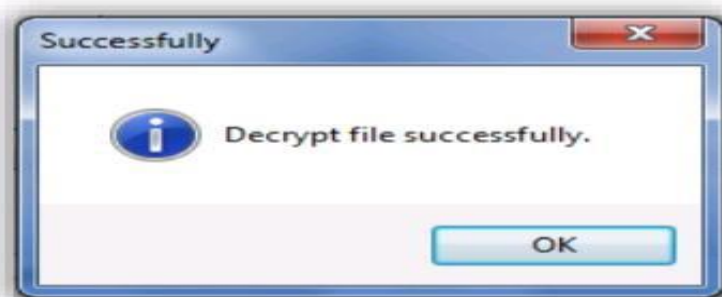


Figure 5-7: Decrypt file successfully message

successfully hide file message box to notify the user of the extract operation is successfully

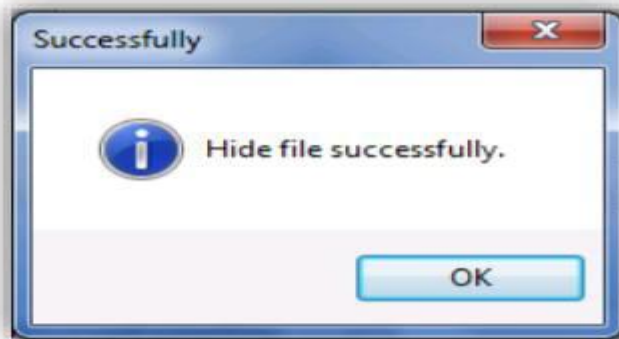


Figure 5-8: Hide file successfully message

SYSTEM IMPLEMENTATION

Step 1: Import all the required python libraries.

```
#import all the required libraries

import cv2
import numpy as np
import types
from google.colab.patches import cv2_imshow #Google colab crashes if you try to display
#image using cv2.imshow() thus use this import
```

Step 2: Define a function to convert any type of data into binary, we will use this to convert the secret data and pixel values to binary in the encoding and decoding phase.

```
[ ] def messageToBinary(message):  
    if type(message) == str:  
        return ''.join([ format(ord(i), "08b") for i in message ])  
    elif type(message) == bytes or type(message) == np.ndarray:  
        return [ format(i, "08b") for i in message ]  
    elif type(message) == int or type(message) == np.uint8:  
        return format(message, "08b")  
    else:  
        raise TypeError("Input type not supported")
```


Step3: Write a function to hide secret message into the image by altering the LSB

```
[ ] # Function to hide the secret message into the image

def hideData(image, secret_message):

    # calculate the maximum bytes to encode
    n_bytes = image.shape[0] * image.shape[1] * 3 // 8
    print("Maximum bytes to encode:", n_bytes)

    #Check if the number of bytes to encode is less than the maximum bytes in the image
    if len(secret_message) > n_bytes:
        raise ValueError("Error encountered insufficient bytes, need bigger image or less data !!")

    secret_message += "#####" # you can use any string as the delimiter

    data_index = 0
    # convert input data to binary format using messageToBinary() function
    binary_secret_msg = messageToBinary(secret_message)

    data_len = len(binary_secret_msg) #Find the length of data that needs to be hidden
    for values in image:
```

```
    for pixel in values:
        # convert RGB values to binary format
        r, g, b = messageToBinary(pixel)
        # modify the least significant bit only if there is still data to store
        if data_index < data_len:
            # hide the data into least significant bit of red pixel
            pixel[0] = int(r[:-1] + binary_secret_msg[data_index], 2)
            data_index += 1
        if data_index < data_len:
            # hide the data into least significant bit of green pixel
            pixel[1] = int(g[:-1] + binary_secret_msg[data_index], 2)
            data_index += 1
        if data_index < data_len:
            # hide the data into least significant bit of blue pixel
            pixel[2] = int(b[:-1] + binary_secret_msg[data_index], 2)
            data_index += 1
        # if data is encoded, just break out of the loop
        if data_index >= data_len:
            break

    return image
```

step4: define a function to decode the hidden message from the stego image

```
[ ] def showData(image):

    binary_data = ""
    for values in image:
        for pixel in values:
            r, g, b = messageToBinary(pixel) #convert the red,green and blue values into binary format
            binary_data += r[-1] #extracting data from the least significant bit of red pixel
            binary_data += g[-1] #extracting data from the least significant bit of red pixel
            binary_data += b[-1] #extracting data from the least significant bit of red pixel

    # split by 8-bits
    all_bytes = [ binary_data[i: i+8] for i in range(0, len(binary_data), 8) ]
    # convert from bits to characters
    decoded_data = ""
    for byte in all_bytes:
        decoded_data += chr(int(byte, 2))
        if decoded_data[-5:] == "####": #check if we have reached the delimiter which is "####"
            break
    #print(decoded_data)
    return decoded_data[:-5] #remove the delimiter to show the original hidden message
```

step 5: Function that takes the input image name and secret message as input from user and calls hide data () to encode the message

```
[ ] # Encode data into image
def encode_text():
    image_name = input("Enter image name(with extension): ")
    image = cv2.imread(image_name) # Read the input image using OpenCV-Python.
    #It is a library of Python bindings designed to solve computer vision problems.

    #details of the image
    print("The shape of the image is: ",image.shape) #check the shape of image to calculate the number of bytes in it
    print("The original image is as shown below: ")
    resized_image = cv2.resize(image, (500, 500)) #resize the image as per your requirement
    cv2.imshow(resized_image) #display the image

    data = input("Enter data to be encoded : ")
    if (len(data) == 0):
        raise ValueError('Data is empty')

    filename = input("Enter the name of new encoded image(with extension): ")
    encoded_image = hideData(image, data) # call the hideData function to hide the secret message into the selected image
    cv2.imwrite(filename, encoded_image)
```

Step 6: Create a function to ask user to enter the name of the image that needs to be decoded and call the showData() function to return the decoded message.

```
# Decode the data in the image
def decode_text():
    # read the image that contains the hidden image
    image_name = input("Enter the name of the steganographed image that you want to decode (with extension) :")
    image = cv2.imread(image_name) #read the image using cv2.imread()

    print("The Steganographed image is as shown below: ")
    resized_image = cv2.resize(image, (500, 500)) #resize the original image as per your requirement
    cv2.imshow(resized_image) #display the Steganographed image

    text = showData(image)
    return text
```

step 7: main function ()

```
# Image Steganography
def Steganography():
    a = input("Image Steganography \n 1. Encode the data \n 2. Decode the data \n Your input is: ")
    userinput = int(a)
    if (userinput == 1):
        print("\nEncoding...")
        encode_text()

    elif (userinput == 2):
        print("\nDecoding...")
        print("Decoded message is " + decode_text())
    else:
        raise Exception("Enter correct input")

Steganography() #encode image
```

Sample code :

```
import cv2
import numpy as np
import types
from google.colab.patches import cv2_imshow

def messageToBinary(message):
    if type(message) == str:
        return ".join([ format(ord(i), "08b") for i in message ])
    elif type(message) == bytes or type(message) == np.ndarray:
        return [ format(i, "08b") for i in message ]
    elif type(message) == int or type(message) == np.uint8:
        return format(message, "08b")
    else:
        raise TypeError("Input type not supported")

def hideData(image, secret_message):

    n_bytes = image.shape[0] * image.shape[1] * 3 // 8
    print("Maximum bytes to encode:", n_bytes)

    if len(secret_message) > n_bytes:
        raise ValueError("Error encountered insufficient bytes, need bigger image or less data !!")

    secret_message += "#####"

    data_index = 0

    binary_secret_msg = messageToBinary(secret_message)

    data_len = len(binary_secret_msg)
    for values in image:

        for pixel in values:
            r, g, b = messageToBinary(pixel)
```

```

    if data_index < data_len:
        pixel[0] = int(r[:-1] + binary_secret_msg[data_index], 2)
        data_index += 1

    if data_index < data_len:

        pixel[2] = int(b[:-1] + binary_secret_msg[data_index], 2)
        data_index += 1

    if data_index >= data_len:
        break

return image

```

```
def showData(image):
```

```

    binary_data = ""
    for pixel in values:
        r, g, b = messageToBinary(pixel)
        binary_data += r[:-1]
        binary_data += g[:-1]
        binary_data += b[:-1]

    all_bytes = [ binary_data[i: i+8] for i in range(0, len(binary_data), 8)]

    decoded_data = ""
    for bytes in all_bytes:
        decoded_data += chr(int(bytes, 2))
        if decoded_data[-5:] == "#####":
            break
    return decoded_data[:-5]

```

```
def encode_text():
```

```

    image_name = input("Enter image name(with extension): ")
    image = cv2.imread(image_name)

    print("The shape of the image is:", image.shape)
    print("The original image is as shown below:")
    resized_image = cv2.resize(image, (500, 500))

```

```

cv2.imshow(resized_image)

data = input("Enter data to be encoded: ")
if(len(data) == 0):
    raise ValueError('Data is empty')

filename = input("Enter the name of new encoded image(with extension): ")
encoded_image = hidenData(image, data)
cv2.imwrite(filename, encoded_image)

def decode_text():

    image_name = input("Enter the name of the steganographed image that you want to
decode(with extension) :")
    image = cv2.imread(image_name)

    print("The steganographed i,age is as shown below: ")
    resized_image = cv2.resize(image, (500, 500))
    cv2.imshow(resized_image)

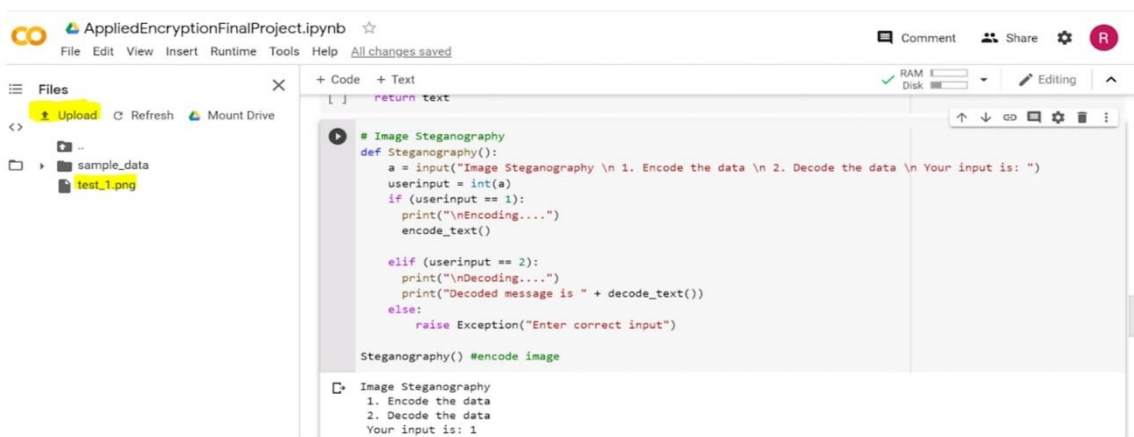
    text = showData(image)

    return text
def steganography():
    a = input("Image Steganography \n 1. Encode the data \n 2. decode the data \n your
input is:")
    userinput = int(a)
    if (userinput == 1):
        print("\nEncoding. . . ")
        encode_text()
    elif (userinput == 2):
        print("\n Decoding. . . ")
        print("Decoded message is " + decode_text())
    else:
        raise Exception("Enter correct input")

Steganography()

```


OUTPUT\RESULTS:



The screenshot displays a Jupyter Notebook titled "AppliedEncryptionFinalProject.ipynb". The interface includes a top menu bar with options like File, Edit, View, Insert, Runtime, Tools, and Help. On the left, a "Files" sidebar shows a directory structure with "sample_data" and "test_1.png". The main area contains a code cell with the following Python code:

```
# Image Steganography
def Steganography():
    a = input("Image Steganography \n 1. Encode the data \n 2. Decode the data \n Your input is: ")
    userInput = int(a)
    if (userInput == 1):
        print("\nEncoding...")
        encode_text()

    elif (userInput == 2):
        print("\nDecoding...")
        print("Decoded message is " + decode_text())
    else:
        raise Exception("Enter correct input")

Steganography() #encode image
```

Below the code cell, the output is displayed:

```
Image Steganography
1. Encode the data
2. Decode the data
Your input is: 1
```

ENCODING THE MESSAGE:

```
Image Steganography
1. Encode the data
2. Decode the data
Your input is: 1
```

```
Encoding....
Enter image name(with extension): test_1.png
The shape of the image is: (1254, 1254, 3)
The original image is as shown below:
```



```
Enter data to be encoded : hakunamatata
Enter the name of new encoded image(with extension): test!_encoded.png
Maximum bytes to encode: 589693
```

DECODING THE MESSAGE:

```
] Image Steganography  
  1. Encode the data  
  2. Decode the data  
Your input is: 2
```

Decoding...

Enter the name of the steganographed image that you want to decode (with extension) :test_!_encoded.png

The Steganographed image is as shown below:



Decoded message is **hakunamatata**

6.SAMPLE TESTING:

According to this classification, software testing is a component of the third phase, and means checking if a program for specified inputs gives correctly and expected results. So the main aim of testing is to analyze the performance and to evaluate the errors that occur when the program is executed with different input sources and running in different operating environments.

There are a many definitions of software testing, but one can shortly define that as: A process of executing a program with the goal of finding errors. So, testing means that one inspects behavior of a program on a finite set of test cases (a set of inputs, execution preconditions, and expected outcomes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement) for which valued inputs always exist.

Testing is an activity performed for evaluating software quality and for improvingg it. Hence, the goal of testing is systematical detection of different classes of errors (error can be defined as a human action that produces an incorrect result) in a minimum amount of time and with a minimum amount of effort.

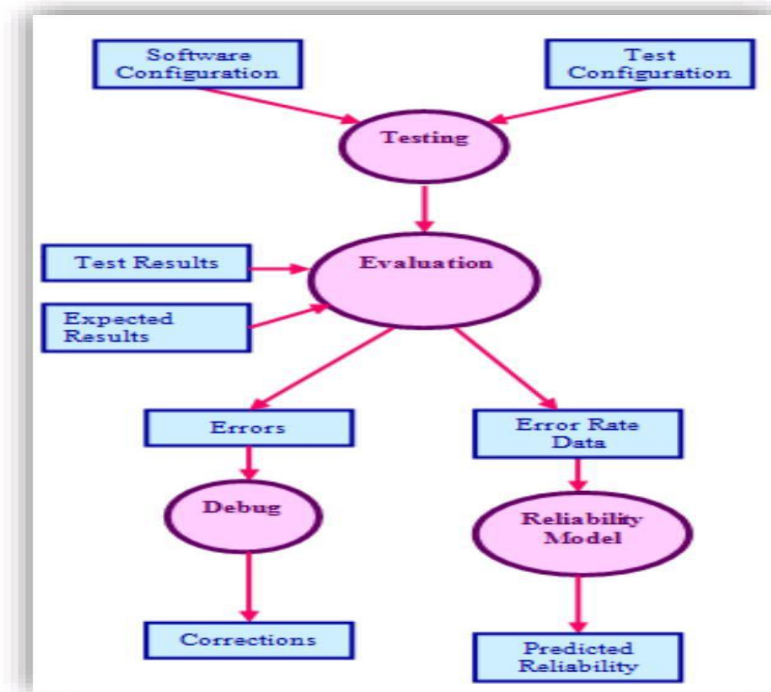


Figure 5-9: Test Information Flow

There are different types of approaches for testing a .NET framework-based application. The types of testing are:

- Unit testing
- Validation testing
- Integration testing
- User acceptance testing
- Output testing
- Black box and white box testing.

METHODS OF TESTING:

Unit testing:

Is the approach of taking a small part of testable application and executing it according to the requirements and testing the application behavior? Unit testing is used for detecting the defects that occur during execution (MSDN, 2010). When an algorithm is executed, the integrity should be maintained by the data structures. Unit testing is made use for testing the functionality of each algorithm during execution. Unit testing can be used in the bottom-up test approach which makes the integration test much easier. Unit testing reduces the ambiguity in the units. Unit testing uses regression testing, which makes the execution simpler. Unit testing has its applications for extreme programming, testing unit frame works and good support for language level unit testing.

Validation Testing: Validation is the process of finding whether the product is built correct or not. The software application or product that is designed should fulfill the requirements and reach the expectations set by the user. Validation is done while developing or at the final stage of development process to determine whether it is satisfies the specified requirements of user.

Using validation test the developer can qualify the design, performance and its operations. Also the accuracy, repeatability, selectivity, Limit of detection and quantification can be specified using Validation testing (MSDN, 2010).

Output Testing: After completion of validation testing the next process is output testing. Output testing is the process of testing the output generated by the application for the specified inputs. This process checks weather the application is producing the required output as per the user's specification or not. The output testing can be done by considering mainly by updating the test plans, the behavior of application with different type of inputs and with produced outputs, making the best use of the operating capacity and considering the recommendations for fixing the issues (MSDN, 2010).

Integration Testing: Integration testing is an extension to unit testing, after unit testing the units are

integrated with the logical program. The integration testing is the process of examining the working behavior of the particular unit after embedding with program. This procedure identifies the problems that occur during the combination of units.

The integration testing can be commonly done in three approaches:

- Top-down approach
- Bottom-up approach
- Umbrella approach

1 Top-down approach: In the top-down approach the highest level module should be considered first and integrated. This approach makes the high level logic and data flow to test first and reduce the necessity of drivers. One disadvantage with top-down approach is its poor support and functionality is limited (MSDN, 2010).

2 Bottom-up approach: Bottom-up approach is opposite to top-down approach. In this approach, the lowest level units are considered and integrated first. Those units are known as utility units. The utility units are tested first so that the usage of stubs is reduced.

The disadvantage in this method is that it needs the respective drivers which make the test complicated, the support is poor and the functionality is limited (MSDN, 2010).

Umbrella approach: The third approach is umbrella approach, which makes use of both the top - bottom and bottom - top approaches. This method tests the integration of units along with its functional data and control paths. After using the top - bottom and bottom-top approaches, the outputs are integrated in top - bottom manner.

The advantage of this approach is that it provides good support for the release of limited functionality as well as minimizing the needs of drivers and hubs. The main disadvantage is that it is less systematic than the other two approaches (MSDN, 2010).

TEST CASES:

STEP	TEST CONDITION	ACTION	EXPECTED RESULT	ACTUAL RESULT
1. Embed File				
1.1	Select Cover Image	Click	Check browse images	Done
1.2	Select Embed Image	Click	Check browse images	Done
1.3	Select Output Image	Click	Check browse images	Done
1.4	Click Embed Button	Click	Check embed function	Done
2. Extract				
2.1	Select Cover Image	Click	Check browse images	Done
2.2	Select Output Image	Click	Check browse images	Done
2.3	Click Extract Button	Click	Check extract function	Done

Table 5-1: Test Cases

CONCLUSION:

It is observed that through LSB Substitution Steganographic method, the results obtained in data hiding are pretty impressive as it utilizes the simple fact that any image could be broken up to individual bit-planes each consisting of different levels of information.

It is to be noted that as discussed earlier, this method is only effective for bitmap images as these involve lossless compression techniques. But this process can also be extended to be used for color images where, bit plane slicing is to be done individually for the top four bit-planes for each of R, G, B of the message image.

BIBLIOGRAPHY

Bailer, W., *wbStego*, available at http://www.8ung.at/wbailer/wbstego/sg_li000.htm, 1998.

Friedman, W.F. and Friedman, E., *The Shakespearean Ciphers Examined*, Syndics of the Cambridge University, U.K., 1958.

Kahn, D., *The Codebreakers*, Macmillan, New York, 1967.

Katzenbeisser, S. and Petitcolas, F.A.P., *Information Hiding: Techniques for Steganography and Watermarking*, Artech House, Boston, 2000.

Kent, P., Art of Anamorphisms, available at <http://www.anamorphosis.com/>.

Low, S.H., Maxemchuk, N.F., Brassil, J.T., and O'Gorman, L., *Document Marking and Identification Using Both Line and Word Shifting*, AT&T Bell Laboratories, New Jersey, 1994.

Schilling, D.L., *Meteor Burst Communications: Theory and Practice*, Wiley Europe, 1993.

Simmons, G.J., "The Prisoners' Problem and the Subliminal Channel," CRYPTO83, *Advances in Cryptology*, August 22–24, 51–67, 1984.

Zim, H.S., *Codes and Secret Writing*, William Morrow, New York, 1948.

REFERENCES:

1. <https://towardsdatascience.com/steganography-hiding-an-image-inside-another-77ca66b2acb1>
2. <https://www.edureka.co/blog/steganography-tutorial>
3. <https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#191d0b0160ba>
4. <https://www.ukessays.com/essays/computer-science/steganography-uses-methods-tools3250.php>
5. <https://www.thepythoncode.com/article/hide-secret-data-in-images-usingsteganography-python>
6. <https://www.youtube.com/watch?v=xepNoHgNj0w&t=1922>